Intro to Language Models: How Generative AI Understands and Responds to Us

Eleni Verteouri

NTUA Athens

November 20, 2024

What Are Language Models?

- AI systems that understand and generate human language.
- Trained on large datasets of text to predict words and phrases.
- Examples of LLM families: GPT, Llama, Gemini and more

What Are Language Models?

- AI systems that understand and generate human language.
- Trained on large datasets of text to predict words and phrases.
- Examples of LLM families: GPT, Llama, Gemini and more

Core Idea: Language models guess the next word based on context.

How Do They Work?

Key Concept: Sequence Probability

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | x_1, \ldots, x_{i-1})$$

How Do They Work?

Key Concept: Sequence Probability

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | x_1, \ldots, x_{i-1})$$

How Do Models Use This?

- Models learn relationships between words by training on large datasets.
- They predict the next word based on the previous ones (context).

How Do They Work?

Key Concept: Sequence Probability

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | x_1, \ldots, x_{i-1})$$

How Do Models Use This?

- Models learn relationships between words by training on large datasets.
- They predict the next word based on the previous ones (context).

Why Is This Useful?

- Helps generate meaningful text by understanding word relationships.
- Enables applications like:
  - Autocomplete (predicting your next word).
  - Text generation (creating new sentences).
  - Translation (understanding word context for accurate results).

Focusing on Key Information: Attention Mechanism

Attention Formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Focusing on Key Information: Attention Mechanism

Attention Formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- Q: Query – Represents what we are focusing on (e.g., a word in the input).
- K: Key – Encodes the context or other words in the input.
- V: Value – Holds the information we want to retrieve.
- d_k: Dimensionality of the key vectors, used to scale the scores.

Focusing on Key Information: Attention Mechanism

Attention Formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- Q: Query – Represents what we are focusing on (e.g., a word in the input).
- K: Key – Encodes the context or other words in the input.
- V: Value – Holds the information we want to retrieve.
- d_k: Dimensionality of the key vectors, used to scale the scores.

How Does It Work?

1. Compute similarity between the query (Q) and the keys (K) by calculating QK^T.
2. Scale the scores by dividing by $\sqrt{d_k}$ to ensure stable gradients.
3. Apply the **softmax function** to convert these scores into probabilities.
4. Use these probabilities to weigh the values (V) and produce the output.

Focusing on Key Information: Attention Mechanism

Attention Formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- Q: Query – Represents what we are focusing on (e.g., a word in the input).
- K: Key – Encodes the context or other words in the input.
- V: Value – Holds the information we want to retrieve.
- d_k: Dimensionality of the key vectors, used to scale the scores.

How Does It Work?

1. Compute similarity between the query (Q) and the keys (K) by calculating QK^T.
2. Scale the scores by dividing by $\sqrt{d_k}$ to ensure stable gradients.
3. Apply the **softmax function** to convert these scores into probabilities.
4. Use these probabilities to weigh the values (V) and produce the output.

Reference: Vaswani et al., Attention Is All You Need, NeurIPS 2017.

Tokenizers

What is a Tokenizer?

- A tokenizer maps a text sequence T into a sequence of tokens $\{t_1, t_2, \ldots, t_n\}$.
- Mathematically:

$$T \rightarrow \{t_1, t_2, \ldots, t_n\}, \quad t_i \in \mathcal{V}$$

where $\mathcal{V}$ is the vocabulary of the tokenizer.

Tokenizers

What is a Tokenizer?

- A tokenizer maps a text sequence T into a sequence of tokens $\{t_1, t_2, \ldots, t_n\}$.
- Mathematically:

$$T \rightarrow \{t_1, t_2, \ldots, t_n\}, \quad t_i \in \mathcal{V}$$

where $\mathcal{V}$ is the vocabulary of the tokenizer.

Objective:

- Minimize the vocabulary size $|\mathcal{V}|$ while maximizing coverage of text.
- Efficiently encode rare words as sequences of subwords.

Tokenizers

What is a Tokenizer?

- A tokenizer maps a text sequence T into a sequence of tokens $\{t_1, t_2, \ldots, t_n\}$.
- Mathematically:

$$T \to \{t_1, t_2, \ldots, t_n\}, \quad t_i \in \mathcal{V}$$

  where $\mathcal{V}$ is the vocabulary of the tokenizer.

Objective:

- Minimize the vocabulary size $|\mathcal{V}|$ while maximizing coverage of text.
- Efficiently encode rare words as sequences of subwords.

Popular Tokenizers: Byte Pair Encoding (BPE), WordPiece

Tokenizers in the Context of LLMs:

- Tokenization is the first step in processing text for an LLM:

$$\text{Raw text } T \to \text{Tokens } \{t_1, t_2, \ldots, t_n\} \to \text{Embeddings}$$

- The model receives token embeddings as inputs, which represent tokens numerically.

Embeddings: Representing Tokens Numerically

What are Embeddings?

- Embeddings are dense vector representations of tokens.
- They map each token $t_i \in \mathcal{V}$ to a high-dimensional numerical vector $\mathbf{e}_i \in \mathbb{R}^d$.

Embeddings: Representing Tokens Numerically

What are Embeddings?

- Embeddings are dense vector representations of tokens.
- They map each token $t_i \in \mathcal{V}$ to a high-dimensional numerical vector $\mathbf{e}_i \in \mathbb{R}^d$.

Why Use Embeddings? Capture semantic relationships:

$$\text{cosine similarity}(\mathbf{e}_{\text{"king"}}, \mathbf{e}_{\text{"queen"}}) \approx \text{cosine similarity}(\mathbf{e}_{\text{"man"}}, \mathbf{e}_{\text{"woman"}})$$

Embeddings: Representing Tokens Numerically

What are Embeddings?

- Embeddings are dense vector representations of tokens.
- They map each token $t_i \in \mathcal{V}$ to a high-dimensional numerical vector $\mathbf{e}_i \in \mathbb{R}^d$.

Why Use Embeddings? Capture semantic relationships:

$$\text{cosine similarity}(\mathbf{e}_{\text{"king"}}, \mathbf{e}_{\text{"queen"}}) \approx \text{cosine similarity}(\mathbf{e}_{\text{"man"}}, \mathbf{e}_{\text{"woman"}})$$

How are Embeddings Used?

- After tokenization, tokens $\{t_1, t_2, \ldots, t_n\}$ are converted to embeddings $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$.
- These embeddings are input to the language model for further processing.

Positional Encoding: Capturing Sequence Order

Why Positional Encoding?

- Transformers lack a natural sense of word order.
- Positional encodings are added to input embeddings to incorporate sequence information.

Positional Encoding: Capturing Sequence Order

Why Positional Encoding?

- Transformers lack a natural sense of word order.
- Positional encodings are added to input embeddings to incorporate sequence information.

Mathematical Representation:

$$\mathrm{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right), \quad \mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

where:

- pos: Position in the sequence.
- i: Dimension index of the embedding vector.
- d: Total embedding dimension.

Positional Encoding: Capturing Sequence Order

Why Positional Encoding?

- Transformers lack a natural sense of word order.
- Positional encodings are added to input embeddings to incorporate sequence information.

Mathematical Representation:

$$\mathrm{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right), \quad \mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

where:

- pos: Position in the sequence.
- i: Dimension index of the embedding vector.
- d: Total embedding dimension.

How Does it Work?

- Positional encodings generate unique patterns for each position in the sequence.
- These patterns are added to the input embeddings:

$$\mathbf{E}_{\text{input}} = \mathbf{E}_{\text{token}} + \mathbf{PE}$$

Cross-Entropy Loss: Training Language Models

What is Cross-Entropy Loss?

- Cross-entropy measures the difference between the predicted probability distribution ($\hat{y}$) and the true distribution (y).
- It is the most commonly used loss function for training language models.

Cross-Entropy Loss: Training Language Models

What is Cross-Entropy Loss?

- Cross-entropy measures the difference between the predicted probability distribution ($\hat{y}$) and the true distribution (y).
- It is the most commonly used loss function for training language models.

Mathematical Formula:

$$\mathcal{L} = -\sum_{i=1}^{n} y_i \log \hat{y}_i$$

where:

- y_i: True probability (1 for the correct token, 0 otherwise).
- $\hat{y}_i$: Predicted probability for token i.

Cross-Entropy Loss: Training Language Models

What is Cross-Entropy Loss?

- Cross-entropy measures the difference between the predicted probability distribution ($\hat{y}$) and the true distribution (y).
- It is the most commonly used loss function for training language models.

Mathematical Formula:

$$\mathcal{L} = -\sum_{i=1}^{n} y_i \log \hat{y}_i$$

where:

- y_i: True probability (1 for the correct token, 0 otherwise).
- $\hat{y}_i$: Predicted probability for token i.

How is it Used?

- At each training step, the model predicts a probability distribution over the vocabulary for the next token.
- Cross-entropy loss is computed by comparing this distribution with the true next token.

Reinforcement Learning with Human Feedback (RLHF)

What is RLHF?

- RLHF is a technique used to improve model alignment with human preferences.
- Combines:
  - Reinforcement learning (RL) for model optimization.
  - Feedback from humans to define what a “good” response looks like.

Reinforcement Learning with Human Feedback (RLHF)

What is RLHF?

- RLHF is a technique used to improve model alignment with human preferences.
- Combines:
  - Reinforcement learning (RL) for model optimization.
  - Feedback from humans to define what a "good" response looks like.

Reinforcement Learning with Human Feedback (RLHF)

What is RLHF?

- RLHF is a technique used to improve model alignment with human preferences.
- Combines:
  - Reinforcement learning (RL) for model optimization.
  - Feedback from humans to define what a "good" response looks like.

Optimization Objective:

$$\max_{\theta} \mathbb{E}[R(f_\theta(x))]$$

where:

- f_θ: the model being optimized,
- x: input data, and
- R: the reward function from the reward model.

Prompt Engineering: Teaching AI Through Inputs

What is Prompt Engineering?

- The process of designing prompts to guide language model outputs.
- Prompts act as instructions or context for the model to generate desired responses.

Prompt Engineering: Teaching AI Through Inputs

What is Prompt Engineering?

- The process of designing prompts to guide language model outputs.
- Prompts act as instructions or context for the model to generate desired responses.

Optimization Goal:

$$\min_{\theta} L(f_\theta(x + p), y)$$

where:

- f_θ: The model,
- x: Input data,
- p: Prompt (custom input), and
- y: Desired output.

Prompt Engineering: Teaching AI Through Inputs

What is Prompt Engineering?

- The process of designing prompts to guide language model outputs.
- Prompts act as instructions or context for the model to generate desired responses.

Optimization Goal:

$$\min_{\theta} L(f_{\theta}(x+p), y)$$

where:

- f_{θ}: The model,
- x: Input data,
- p: Prompt (custom input), and
- y: Desired output.

Why Prompt Engineering?

- Enhances the model's ability to follow specific instructions.
- Reduces errors or irrelevant responses by providing clearer context.
- Widely used in applications like zero-shot and few-shot learning.

One-Shot and Few-Shot Learning

What is One-Shot Learning?

- The model learns to perform a task from a single example.
- Requires strong generalization capabilities.

One-Shot and Few-Shot Learning

What is One-Shot Learning?

- The model learns to perform a task from a single example.
- Requires strong generalization capabilities.

What is Few-Shot Learning?

- The model learns from a small number of examples (typically 2-5).
- Allows LLMs to adapt to new tasks without extensive fine-tuning.

One-Shot and Few-Shot Learning

What is One-Shot Learning?

- The model learns to perform a task from a single example.
- Requires strong generalization capabilities.

What is Few-Shot Learning?

- The model learns from a small number of examples (typically 2-5).
- Allows LLMs to adapt to new tasks without extensive fine-tuning.

Mathematical Representation:

$$P(y|x, \{(x_1, y_1), \ldots, (x_k, y_k)\})$$

where:

- x: New input.
- $\{(x_1, y_1), \ldots, (x_k, y_k)\}$: Few examples.
- y: Predicted output based on the context of examples.

One-Shot and Few-Shot Learning

What is One-Shot Learning?

- The model learns to perform a task from a single example.
- Requires strong generalization capabilities.

What is Few-Shot Learning?

- The model learns from a small number of examples (typically 2-5).
- Allows LLMs to adapt to new tasks without extensive fine-tuning.

Mathematical Representation:

$$P(y|x, \{(x_1, y_1), \ldots, (x_k, y_k)\})$$

where:

- x: New input.
- $\{(x_1, y_1), \ldots, (x_k, y_k)\}$: Few examples.
- y: Predicted output based on the context of examples.

Why is it Important?

- Reduces the need for large labeled datasets.
- Enables models to handle novel tasks dynamically.

Chain-of-Thought Prompting

What is Chain-of-Thought Prompting?

- A technique to guide LLMs through intermediate reasoning steps before generating a final answer.

Chain-of-Thought Prompting

What is Chain-of-Thought Prompting?

- A technique to guide LLMs through intermediate reasoning steps before generating a final answer.

Mathematical Representation:

$$P(y|x, z_1, z_2, \ldots, z_k)$$

where:

- x: Input query.
- $\{z_1, z_2, \ldots, z_k\}$: Intermediate reasoning steps.
- y: Final output.

Chain-of-Thought Prompting

What is Chain-of-Thought Prompting?

- A technique to guide LLMs through intermediate reasoning steps before generating a final answer.

Mathematical Representation:

$$P(y|x, z_1, z_2, \ldots, z_k)$$

where:

- x: Input query.
- $\{z_1, z_2, \ldots, z_k\}$: Intermediate reasoning steps.
- y: Final output.

Why is it Important?

- Improves reasoning and accuracy for complex tasks.
- Allows LLMs to handle multi-step problems effectively.

Hyperparameter: Temperature

What is Temperature?

- Controls the randomness of token selection in LLM output.
- Scales the logits (model's raw token scores) before applying softmax.

Hyperparameter: Temperature

What is Temperature?

- Controls the randomness of token selection in LLM output.
- Scales the logits (model's raw token scores) before applying softmax.

Mathematical Representation:

$$P(t|x) = \mathrm{softmax}\left(\frac{z}{T}\right)$$

where:

- z: Logits (model's raw scores for each token).
- T: Temperature hyperparameter.

Hyperparameter: Temperature

What is Temperature?

- Controls the randomness of token selection in LLM output.
- Scales the logits (model's raw token scores) before applying softmax.

Mathematical Representation:

$$P(t|x) = \text{softmax}\left(\frac{z}{T}\right)$$

where:

- z: Logits (model's raw scores for each token).
- T: Temperature hyperparameter.

Effect of Temperature:

- $T \to 0$: Results become almost deterministic.
  - Only the highest-probability token is selected.
- $T = 1$: Standard sampling.
- $T > 1$: Increases randomness, sampling from lower-probability tokens.

Hyperparameter: Temperature

What is Temperature?

- Controls the randomness of token selection in LLM output.
- Scales the logits (model's raw token scores) before applying softmax.

Mathematical Representation:

$$P(t|x) = \text{softmax}\left(\frac{z}{T}\right)$$

where:

- z: Logits (model's raw scores for each token).
- T: Temperature hyperparameter.

Effect of Temperature:

- $T \to 0$: Results become almost deterministic.
  - Only the highest-probability token is selected.
- $T = 1$: Standard sampling.
- $T > 1$: Increases randomness, sampling from lower-probability tokens.

Applications:

- Low T: Suitable for factual queries or tasks requiring precision.
- High T: Useful for creative tasks like poetry or storytelling.

Hyperparameter: Top-p (Nucleus Sampling)

What is Top-p?

- Controls the breadth of vocabulary in token selection.
- Filters tokens based on cumulative probability.

Hyperparameter: Top-p (Nucleus Sampling)

What is Top-p?

- Controls the breadth of vocabulary in token selection.
- Filters tokens based on cumulative probability.

How Does It Work?

- Sort tokens by their probabilities.
- Retain only the top p fraction of cumulative probability:

$$\sum_{t \in \text{top-p}} P(t|x) \geq p$$

- Discard tokens beyond this threshold.

Hyperparameter: Top-p (Nucleus Sampling)

What is Top-p?

- Controls the breadth of vocabulary in token selection.
- Filters tokens based on cumulative probability.

How Does It Work?

- Sort tokens by their probabilities.
- Retain only the top p fraction of cumulative probability:

$$\sum_{t \in \text{top-p}} P(t|x) \geq p$$

- Discard tokens beyond this threshold.

Effect of Top-p:

- Low p: Narrow selection of high-probability tokens (more focused).
- High p: Broader selection of tokens (more diverse outputs).

Hyperparameter: Top-p (Nucleus Sampling)

What is Top-p?

- Controls the breadth of vocabulary in token selection.
- Filters tokens based on cumulative probability.

How Does It Work?

- Sort tokens by their probabilities.
- Retain only the top p fraction of cumulative probability:

$$\sum_{t \in \text{top-p}} P(t|x) \geq p$$

- Discard tokens beyond this threshold.

Effect of Top-p:

- Low p: Narrow selection of high-probability tokens (more focused).
- High p: Broader selection of tokens (more diverse outputs).

Applications:

- Low p: Useful for tasks requiring coherent, precise outputs.
- High p: Encourages creative and exploratory outputs.

Retrieval-Augmented Generation (RAG): Enhancing Outputs

Definition: RAG enhances LLM outputs by incorporating external knowledge into the generation process.

Retrieval-Augmented Generation (RAG): Enhancing Outputs

Definition: RAG enhances LLM outputs by incorporating external knowledge into the generation process.

Steps in RAG:

1. **Query Embedding:** $q = E_q(x)$
2. **Document Retrieval:** $D = \{d_1, d_2, ..., d_k\}$
3. **Scoring:**
$$\text{score}(q, d) = \cos(E_q(q), E_d(d))$$
4. **Augmented Input:** $x_{aug} = [x; D]$
5. **Generation:** $y = f_\theta(x_{aug})$

Retrieval-Augmented Generation (RAG): Enhancing Outputs

Definition: RAG enhances LLM outputs by incorporating external knowledge into the generation process.

Steps in RAG:

1. **Query Embedding:** $q = E_q(x)$
2. **Document Retrieval:** $D = \{d_1, d_2, ..., d_k\}$
3. **Scoring:**
$$\text{score}(q, d) = \cos(E_q(q), E_d(d))$$
4. **Augmented Input:** $x_{aug} = [x; D]$
5. **Generation:** $y = f_\theta(x_{aug})$

Key Components:

- E_q, E_d: Embedding functions for queries and documents
- D: Retrieved documents
- x_{aug}: Augmented input combining original query and retrieved information

Fine-Tuning: Specializing LLMs

Definition: Fine-tuning adapts a pre-trained LLM to specific tasks or domains by updating its parameters using task-specific data.

Fine-Tuning: Specializing LLMs

Definition: Fine-tuning adapts a pre-trained LLM to specific tasks or domains by updating its parameters using task-specific data.

Optimization Objective:

$$\min_{\theta} L(f_\theta(x), y)$$

where:

- L: Loss function (e.g., cross-entropy)
- x: Input data
- y: Target output

Fine-Tuning: Specializing LLMs

Definition: Fine-tuning adapts a pre-trained LLM to specific tasks or domains by updating its parameters using task-specific data.

Optimization Objective:

$$\min_{\theta} L(f_\theta(x), y)$$

where:

- L: Loss function (e.g., cross-entropy)
- x: Input data
- y: Target output

Parameter Update:

$$\theta_{new} = \theta_{old} - \alpha \nabla_\theta L(f_\theta(x), y)$$

where:

- α: Learning rate

Fine-Tuning: Specializing LLMs

Definition: Fine-tuning adapts a pre-trained LLM to specific tasks or domains by updating its parameters using task-specific data.

Optimization Objective:

$$\min_{\theta} L(f_\theta(x), y)$$

where:

- L: Loss function (e.g., cross-entropy)
- x: Input data
- y: Target output

Parameter Update:

$$\theta_{new} = \theta_{old} - \alpha \nabla_\theta L(f_\theta(x), y)$$

where:

- α: Learning rate

Why Fine-Tune?

- Adapts the model to domain-specific tasks
- Retains general knowledge from pre-training while focusing on

Why It Matters

- Language models are transforming how we interact with AI.
- Accessible tools like ChatGPT make AI more user-friendly.
- Understanding the basics helps in leveraging their full potential.

Why It Matters

- Language models are transforming how we interact with AI.
- Accessible tools like ChatGPT make AI more user-friendly.
- Understanding the basics helps in leveraging their full potential.

Takeaway: You don't need to be an expert to start using AI effectively!

Questions?

Thank you for attending!

Feel free to ask questions or share your thoughts.